Journal of Engineering Science and Technology
Vol. 13, No. 4 (2018) 1134 - 1142
© School of Engineering, Taylor's University

PROPAGATION OF MISCLASSIFIED INSTANCES TO HANDLE NONSTATIONARY IMBALANCED DATA STREAM

MEENAKSHI A. THALOR*, S. T. PATIL

Vishwakarma Institute of Technology, Savitribai Phule Pune University, Pune, India
*Corresponding Author: thalor.meenakshi@gmail.com

Abstract

Learning on the data stream with nonstationary and imbalanced property is an interesting and complicated problem in data mining as change in class distribution may result in class unbalancing. Many real time problems like intrusion detection, credit card fraud detection, weather forecasting and many more applications suffer concept drift as well as class imbalance as they change with time. The rationale of this paper is to present an effective learning for nonstationary imbalanced data stream which emphasis on misclassified examples with the focus on two-class problems. At the end of paper, proposed algorithms is compared with existing similar approaches using various evaluation metrics.

Keywords: Bagging, Class imbalance, Data stream ensemble, Nonstationary data.

1. Introduction

A data stream is a well-organized sequence of data examples/blocks that continually arrive at a specific rate, potentially unbounded in size that is once data examples/block is processed, it is usually discarded. Subsequently, data examples/ blocks can be generated from different sources which causes continuous evolving pattern and burstiness in data. All these characteristics make the data stream impossible to process by traditional data mining approaches. The data stream classification algorithms should be incremental in nature which will read data examples at a time t, rather than acquiring all the data at the beginning. It should make only one look into the data, should process each instance at a constant time and use a constant amount of memory and should be ready to predict at any time.

Nonstationary data is a time series data where data continuously evolve over the time; for all values and every time period time series does not preserve the constant mean, variance, and autocorrelation structure. Figure 1 depicts supervised

Nomenclatures

D^t	Dataset available at time t
Sig_k^t	Sigmoid Weight of k[th] base classifier at time t
w_k^t	Weight of k[th] base classifier at time t

Greek Symbols

ε_k^t	Error rate of k[th] sub-ensemble at time t

Abbreviations

ENSIDS	Ensemble for Nonstationary Imbalanced Data Stream
F-M	F-measure
G-M	G-mean
SMOTE	Synthetic Minority Oversampling Technique
WMV	Weighted Majority Weighting

learning on a nonstationary data stream where the data source at training time is not equivalent to the data source at the testing time. Ensemble based classification has proved its significance to handle nonstationary data stream where it groups the classifiers by some way, then individual predictions of each classifier is combined to classify unseen data.

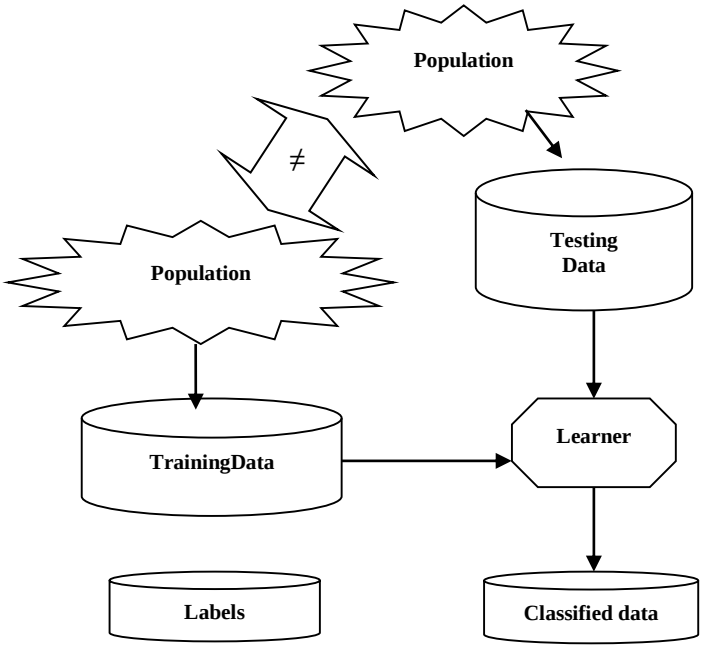

Fig. 1. Supervised learning on nonstationary data stream.

The remainder of this paper is presented as follows. The subsequent section presents related work. A learning algorithm for nonstationary and imbalanced data using misclassified instances is described in Section 3. Section 4 describes comparative evaluation and analysis. Finally, Section 5 concludes with future work.

2. Related Work

In literature, handling nonstationary data and class imbalanced issue is solved independently. Few researchers have considered these two issues simultaneously. The first experiment of ensembles on data streams was Streaming Ensemble Algorithm [1] which uses a batch of d instances to build a classifier, if current classifier improves the results of ensemble, it gets included at the cost of the worst classifier. Accuracy Weighted Ensemble [2] make classifiers on each incoming data batch. The weight to each classifier is given based on mean square error. Accuracy Updated Ensemble [3] derived from AWE, which makes new classifiers on new chunk and conditionally updates existing classifiers on same chunk. Learn++.NSE [4] generates classifiers sequentially, first converts the incoming data stream into a series of batches of a fixed size and classifier's weight is assigned based on prediction error on previous and current batch.

Accuracy Updated Ensemble 2 [5] introduces unconditional updating of classifiers which is effective for sudden and gradual changes. It is a new weighting mechanism where there is no need of cross-validation on created classifiers and a classifier buffer. A complete review and experiment on ensemble based approaches to handle non stationary data is presented in [6-8].

Class imbalanced problem is solved in the literature by different means like resampling methods, ensemble learning approach, cost sensitive learning and modification of existing learning algorithms. In general, algorithm level and cost-sensitive approaches are more dependent on the problem, whereas sampling and ensemble learning methods are more adaptable since they can be used independently of the base classifier [9]. Here the survey of bagging based ensemble learning to solve class imbalanced problem is presented.

Bagging (Bootstrap AGGregatING) is ensemble based approach to combine classifiers, by providing the different set of inputs using sampling with replacement. Finally the prediction is given by a majority voting which ensures the decrease in variance and errors are ignored. Asymmetric bagging [10], the full minority class is preserved and then the equal size of majority class is considered in each partition in every bootstrap iteration.

SMOTEBagging [11], firstly uses SMOTE [12] where more minority class examples are synthetically generated, and then bagging is applied to majority class examples. UnderBagging, each subset is created by under-sampling majority classes randomly to construct the kth classifiers. In the similar way OverBagging forms, each subset simply by over-sampling minority classes randomly. Roughly balanced bagging [13] do maintain a balance between majority and minority class by assigning weights to instances in each bootstrap iteration. Lazy Bagging [14] uses a nearest neighbor algorithm then applies bagging only on the k nearest points. Learn^{++}.NIE and Learn^{++}.CDS algorithms simultaneously address nonstationary and imbalanced data stream [15] but both these algorithms are combination of existing approaches. Learn^{++}.CDS is a combination of SMOTE and Learn^{++}.NSE, while Learn^{++}.NIE is a combination of Bagging Variation [16] and Learn^{++}.NSE.

3. Ensemble for Nonstationary Imbalanced Data Stream (ENSIDS)

ENSIDS is extended work on Learn^{++}.NSE algorithm which was purely designed for nonstationary data only. Existing algorithms address nonstationary and

imbalanced issue separately and some of the algorithms those address nonstationary and imbalanced issue simultaneously uses instance weighting mechanism. For a big dataset, instance weighting and updating of weight of each instance, every time is not feasible. ENSIDS is proposed algorithm which can be used for the nonstationary environment as well as for the imbalanced environment as shown in Fig. 2. In our approach all instances are equally important while they are employing for training so uniform weight is considered. Secondly, we are propagating the false positive and false negative observations of a classifier to subsequent classifier for improving the performance.

Initially from nonstationary and imbalanced data stream, the batch formation is carried out. After getting the chunk of data available at time t, we are applying bagging technique and multiple bags of chunk is created. For each chunk of bag, a classifier is trained and tested on current data and misclassified instances of each classifier is propagated to the subsequent classifier for training along with its bag chunk. The propagation of misclassified instances to the next classifier in the sub-ensemble is only carried to improve the performance of the algorithm. In addition, based on the performance of each classifier a weight is assigned to them and this weight gets updated when a classifier evaluates the incoming data. The final prediction is carried out by WMV.

In step 1, for each batch of data at time t, a sub-ensemble is created by using BaggingPropagation. In Bagging Propagation, initially, examples are randomly selected to form current bag chunk and from this chunk a hypothesis is created. After formation of the first hypothesis, we have checked the validity of generated hypothesis using current chunk and all misclassified instances is preserved in the buffer and these misclassified instances are added in next bag chunk.

After the formation of the sub-ensemble as in step 1, the performance of existing sub-ensembles will be evaluated on D^t and will get ε_k^t which is error rate of kth sub-ensemble on current D^twhich is computed from condition 1-(F-M). If error generated by the current sub-ensemble is more than 0.5 that is half of the predictions are wrong, and then generate a new sub-ensemble for the current distribution. If error generated by one of the previous sub-ensemble is more than .5 then set its $\varepsilon_k^t = 0.5$ as in step 2.

In step 3, we are assigning weights to create classifiers using a nonlinear sigmoid function. Because of this, if a classifier will be evaluated more than once, then its sigmoid weight will be increased. The weight to the classifier is assigned based on its performance on previous distributions as well as on recent distribution, so weighted average of classifier is computed in step 3. When a classifier is generated, its initial weight is set to 1, after its evaluation on recent environment its w_k^t gets keep updated. The weight of classifier is decreased if it does not perform well on recent distribution, otherwise it remains same. If a classifier's prediction is not correct on recent environment, its weighted error $(w_k^t . \varepsilon_k^t)$ gets increased.

In step 4 the weight error average is computed to determine the voting weight of classifiers. The voting power of each classifier is computed using logarithm of the inverse of its weighted error average. If weighted error average is high, a classifier will get less power of voting. Step 5 gives final prediction on unseen data by using weighted majority voting.

Input: For each dataset D^t where t=1, 2, …..
Training data: $\{x_i^t \in X;\ y_i^t \in Y = \{1, \ldots, c\}\}, i = 1 \ldots\ldots m$ instances
Sigmoid parameters: a &b
Error weight: 0 to 1
Description: Supervised learning algorithm for Nonstationary Imbalanced data stream

Pseudo code:
Do for t=1, 2…
1. Call H_t=BaggingPropagation(BaseClassifier,D^t,T)
2. Evaluate all existing sub-ensembles on D^t using F-M

$$\varepsilon_k^t\text{=1-F-M}$$

If $\varepsilon_{k=t}^t > \frac{1}{2}$ generate a new $\text{sub} - \text{ensemble}$

If $\varepsilon_{k<t}^t > \frac{1}{2}$ set $\varepsilon_k^t = 1/2$

3. Compute the weighted for classifier h_k: for a, b ϵ R

$$\text{Sig}_k^t = \frac{1}{1 + e^{-a(t-k-b)}} \quad ,$$

$$w_k^t = \left\{ \begin{matrix} 1 & t = k \\ \dfrac{\text{Sig}_k^t}{\text{Sig}_k^t + \sum_{j=1}^{t-1} w_k^{t-j}} & \text{otherwise} \end{matrix} \right\}$$

$$\bar{\beta}_k^t = \sum_{j=1}^{t} w_k^j \, \varepsilon_k^j \text{ for } k = 1, \ldots, t$$

4. Calculate classifier voting weights

$$\text{Voting } w_k^t = \log(1/\bar{\beta}_k^t) \text{ for k=1 to t}$$

5. Obtain the final hypothesis

$$H^t(x) = \arg\max_{c \epsilon Y} \sum_{k=1}^{T} w_k^t \, [|h_k(x) = c|]$$

BaggingPropagation(BaseClassifier,D^t,T)
Given training data D^t=$x_i^t \in X;\ y_i^t \in Y$ where i=1,2,…m
T is no. of classifiers to generate

For k =1,2…T
1. If k=1
   Form dataset S^k by selecting *m* random examples from D^t
   Else
   Form dataset S^k by selecting *m* random examples from D^t and add misclassified instances provided by h_{k-1}
2. Call base classifier with S^k to form a hypothesis h_k:X->Y

End For
Composite Hypothesis for unlabeled instance x

$$H^t(x) = \arg\max_{c \epsilon Y} \sum_{k=1}^{T} \frac{1}{T} [|h_k(x) = c|]$$

Fig. 2. The mathematical model of the ENSIDS.

4. Comparative Evaluation And Analysis

For evaluation and analysis, algorithms are implemented in Java using MOA and WEKA libraries and tools. In this section, we depict the datasets and experiment results.

4.1. Datasets

For comparison of ENSIDS with Learn^{++}.NSE and Learn^{++}.NIE, Naïve Bayes classifier, different timestamps, different evaluation measures [8] and no pruning strategy are considered. ENSIDS is evaluated over various real time datasets, as these are nonstationary and imbalanced and Table 1 shows the brief summary of the tested dataset with their imbalanced ratio.

- Diabetes Dataset: This dataset is the Pima Indian diabetes dataset from the UCI Machine Learning Repository which consists of 768 examples. In this dataset, we are considering eight attributes to predict positive and negative class.
- Stock Market Dataset: In stock data, we are considering open, high, low, close, volume and rate of change in closing price to find out Stock index movement i.e. up and down. For training purpose, data from period 2-Jan-2000 to 25-Apr-2016 is fetched and for testing purpose, data from period 2-Jan-2001 to 25-Apr-2016 is fetched using Google finance. Here we are considering IBM, IBN and INFY stock data.
- Airlines Dataset: The dataset consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. Here we have considered only the airline data of year 1989 which is converted into two datasets as airlines_s and airlines_l.

Experiments are done on above mentioned dataset with different imbalanced ratio, to rigorously evaluate ENSIDS.

Table 1. Tested dataset and its summary.

Dataset	Source	#Ex	#Maj.	#Min.	Imbalance Ratio
Diabetes	UCI machine Learning Repository	768	500	268	1.86
Airlines_s	Data Expo 2009	5000	4015	985	4.07
Airlines_l	Data Expo 2009	686503	597229	89274	6.68
IBM_stock	Google Finance	3850	3083	767	4.01
IBN_stock	Google Finance	3848	2588	1260	2.05
INFY_stock	Google Finance	3805	2630	1175	2.23

4.2. Comparative analysis

Table 2 illustrates the comparison results of ENSIDS with the state of art approaches using F-M and G-M evaluation metric on Diabetes dataset. For every different timestamp performance of ENSIDS is better and maintaining a good balance between precision and recall.

Table 2. Evaluation results on diabetes dataset.

Algorithms	Timestamp	F-M	G-M
Learn^{++}.NSE	50	59.45	73.04
Learn^{++}.NIE		62.06	72.22
ENSIDS		59.88	72.46
Learn^{++}.NSE	100	63.45	71.26
Learn^{++}.NIE		62.28	72.76
ENSIDS		63.14	74.23
Learn^{++}.NSE	150	62.60	72.51
Learn^{++}.NIE		63.56	74.32
ENSIDS		63.80	74.92

Table 3 shows comparison results of ENSIDS with the state of art approaches using F-M and G-M evaluation metric on Airlines dataset. For every different timestamp, the performance of true positive and true negative rates of ENSIDS is high as compare to others.

Table 3. Evaluation results on airlines dataset.

Algorithm	Airlines_S			Airlines_L		
	Timestamp	F-M	G-M	Timestamp	F-M	G-M
Learn^{++}.NSE	500	83.07	91.37	60000	79.72	85.79
Learn^{++}.NIE		88.42	92.56		80.71	86.24
ENSIDS		88.36	94.28		84.03	90.07
Learn^{++}.NSE	600	66.62	70.98	75000	83.13	88.96
Learn^{++}.NIE		88.65	92.93		80.60	86.25
ENSIDS		89.42	93.96		83.62	89.58
Learn^{++}.NSE	700	84.63	86.85	100000	80.89	85.69
Learn^{++}.NIE		89.09	92.30		79.80	85.37
ENSIDS		89.28	94.37		83.54	89.83

Table 4 shows comparison results of the proposed algorithm with the existing similar approaches using F-M and G-M evaluation metric on Stock dataset. The choice of optimal timestamp differs from dataset to dataset, hence performance varies.

Table 4. Evaluation results on stock dataset.

Algorithm	Time stamp	IBM_stock		IBN_stock		INFY_stock	
		F-M	G-M	F-M	G-M	F-M	G-M
Learn^{++}.NSE	300	38.45	91.19	72.08	90.81	72.91	75.71
Learn^{++}.NIE		60.04	93.57	84.45	93.58	71.16	91.29
ENSIDS		66.84	94.33	84.79	94.09	49.07	87.65
Learn^{++}.NSE	400	90.87	96.31	86.08	93.60	56.04	79.55
Learn^{++}.NIE		84.36	92.04	83.19	86.70	80.92	84.73
ENSIDS		87.75	96.00	91.78	95.60	84.11	86.84
Learn^{++}.NSE	500	66.78	91.61	78.69	91.55	85.37	87.81
Learn^{++}.NIE		65.56	93.94	89.10	95.29	76.89	81.14
ENSIDS		70.47	94.24	88.32	95.19	90.06	96.08
Learn^{++}.NSE	600	71.82	93.64	84.24	90.65	85.75	88.07
Learn^{++}.NIE		82.21	95.58	88.98	95.18	78.53	82.55
ENSIDS		83.96	96.20	90.78	95.90	90.40	94.10

Figure 3 shows the graphical representation the results of ENSIDS on all evaluated stock dataset which depicts the performance of ENSIDS as compare to existing approaches.

Results on different datasets show that, the performance of ENSIDS is better as compare to Learn^{++}.NSE and Learn^{++}.NIE on F-measure and G-mean evaluation measures.

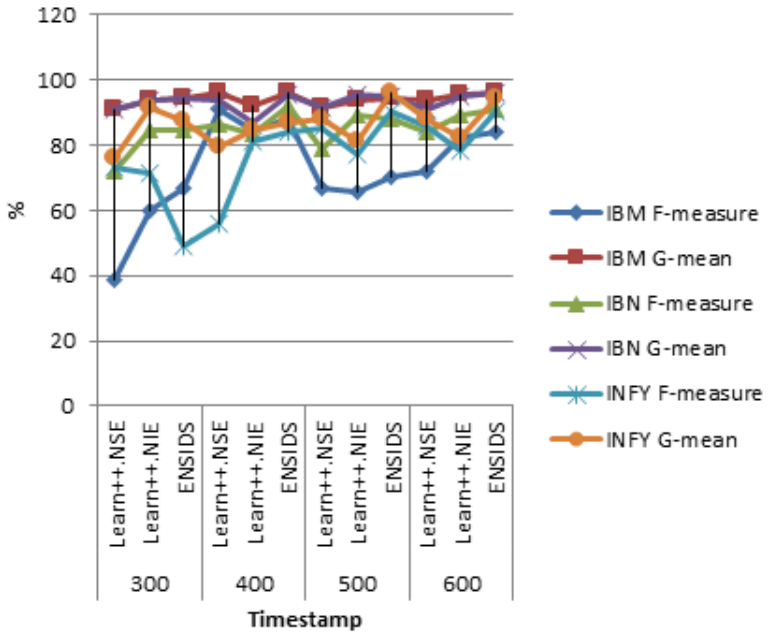

Fig. 3. Graph for stock dataset.

5. Conclusion and Future Work

The proposed work is compared to existing algorithms on different dataset. On every dataset different timestamps are applied, to observe the performance of algorithms. The choice of optimal timestamp differs from dataset to dataset. Comparison results ensure the validity and superiority of proposed work on F-measure and G-mean parameters where ENSIDS performs better as compare to existing approaches. The drift detection mechanism can also be added in the proposed work as a future enhancement. Some of the further enhancements would be to implement the same approach for a parallel computing platform which will reduce the time required for the processing big data.

References

1. Street, W.N.; and Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification, *Intelligent Conference on Knowledge Discovery & Data Mining*, 377-382.
2. Wang, H.; Fan, W.; and Yu, P.; and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. *In Proceedings of the ninth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, 226-235.

3. Brzezinski, D.; and Stefanowski, J. (2011). Accuracy updated ensemble for data streams with concept drift. *Proceedings of the 6th International Conference on Hybrid Artificial Intelligent Systems*, Volume Part II, Wroclaw, Poland, 155-163.
4. Muhlbaier, M.; and Polikar, R. (2007).Multiple classifiers based incremental learning algorithm for learning in nonstationary environments. *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, 6, 3618-3623.
5. Brzezinski, D.; and Stefanowski, J. (2014). Reacting to different types of concept drift: The Accuracy Updated Ensemble Algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1), 81-94
6. Thalor, M.A.; and Patil, S.T. (2014). Review of ensemble based classification algorithms for nonstationary and imbalanced data. *IOSR Journal of Computer Engineering*, 16, 103-107.
7. Thalor, M.A.; and Patil, S.T. (2016). Learning on high frequency stock market data using misclassified instances in ensemble. *International Journal of Advanced Computer Science and Applications*, 7(5), 283-288.
8. Thalor, M.A.; and Patil, S.T. (2016). Incremental learning on nonstationary data stream using ensemble approach. *International Journal of Electrical and Computer Engineering*, 6(4), 1811-1817.
9. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; and Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, MAN and Cybernetics* 42(4), 463-484.
10. Tao D.; Tang X.; Li, X.; and Wu, X. (2006).Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7), 1088-1099.
11. Wang S.; and Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. *In Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*.
12. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; and Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence*, 16, 321-357.
13. Hido, S.; and Kashima, H. (2008). Roughly balanced bagging for imbalanced data. *In SIAM International Conference on Data Mining (SDM)*, 143-152.
14. Zhu X.; and Yang, Y. (2008). A lazy bagging approach to classification. *Pattern Recognition*, 41(10), 2980-2992.
15. Ditzler, G.; and Polikar, R. (2013).Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, 25(10), 2283-2301.
16. Li, C. (2007). Classifying imbalanced data using a bagging ensemble variation. *In ACM-SE 45: Proceedings of the 45th Annual Southeast Regional Conference*, 203-208.